STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AIM-124

SPEECH ANALYSIS BY CLUSTERING, OR THE HYPERPHONEME METHOD

BY

M. M. ASTRAHAN

COMPUTER SCIENCE DEPARTMENT

STANFORD UNIVERSITY

24

STANFORD ARTIFICIAL INTELLIGENCE PROJECT MAY 1970
MEMO AIM-124

SPEECH ANALYSIS BY CLUSTERING, OR THE HYPERPHONEME METHOD

by

M. M. Astrahan

ABSTRACT: In this work, measured speech waveform data was used as a basis for partitioning an utterance into segments and for classifiying those segments. Mathematical classifications were used instead of the traditional phonemes or linguistic categories. This involved clustering methods applied to hyperspace points representing periodic samples of speech waveforms. The cluster centers, or hyperphonemes (HPs), were used to classify the sample points by the nearest-neighbor technique. Speech segments were formed by grouping adjacent points with the same classification. A dictionary of 54 different words from a single speaker was processed by this method. 216 utterances, representing four more repetitions by the same speaker each of the original 54 words, were similarly analyzed into strings of hyperphonemes and matched against the dictionary by heuristically developed formulas. 87% were correctly recognized, although almost no attempt was made to modify and improve the initial methods and parameters.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Department of Defense (SD-183). The work was done at the Artificial Intelligence Project of the Stanford University Computer Science Department.

INTRODUCTION

This work was performed by the author at the Stanford University Computer Science Department Artificial Intelligence Project during a sabbatical year granted by the IBM Advanced Systems Development Division. The work was done with the advice and counsel of Drs. A. L. Samuel and D. R. Reddy. Generous assistance was provided by Gary Goodman, Lee Erman and Ken Siberz in particular and by many members of the project staff in general. An important point of reference was a speech recognition system developed by P. Vicens and D. Reddy. It is operational at the A. I. Project and is described in Vicens' doctoral dissertation[8].

GOALS

There were two goals for this work. The first was to develop an algorithmic learning technique for segmenting and classifying speech utterances, minimizing dependence on heuristic methods and leading to a compact dictionary representation of reference utterances. The second goal was to develop a method of identifying unknown utterances by comparing them with the reference utterances.

BACKGROUND

The Reddy-Vicens system[8] partitions an utterance into segments by means of heuristic formulas applied to the differences between successive periodic samples of the speech waveform. In general, successive samples which exhibit small intersample differences are grouped into segments. Classification formulas are then used to place the segments into 5 phonetic categories. For reference utterances, the classification and time duration of each segment are stored, along with a set of primary data measurements for each segment typical of the samples which were grouped. This data is used for subsequent comparison with segments of utterances to be identified. In the comparison of an unknown and a reference utterance, corresponding segments are determined from the classifications and then their durations and data measurements are compared.

A classification in terms of standard phonemes or phoneme group categories might result in a more compact reference representation by eliminating the need to store the typical measured data for each segment. An unknown utterance would then be identified from its segment classifications and durations. An attempt was made to apply such classifications to the Reddy-Vicens segments by standard pattern matching techniques, using a training set of data measurements corresponding to standard phonemes. This attempt failed because of wide variations in the training set among the data representing a given phoneme. This led to doubts about the feasibility of characterizing classical phonemes in terms of speech data measurements, at least in terms of the measurements being used. It seemed more reasonable to let the measurements themselves identify phoneme-like classes by means of clustering analysis.

DATA

The methods described in this report were tested on the list of 54 computer-related words from Gold[5], shown in Figure 1. The list was recorded by Dr. K. Stevens at Cambridge, Massachusetts, and provided to the Project by Dr. D. Bobrow. The preprocessing hardware will be described only briefly since it is described in detail in Vicens' dissertation[8]. Only three channels were used, corresponding roughly to the frequency range of the first two formats and to the higher frequencies. The filters were 150 Hz. to 900 Hz., 900 Hz. to 2200 Hz., and 2200 Hz. to 5000 Hz. Readings were taken for each channel, every ten milliseconds, of the peak-to-peak amplitude and the number of zero crossings for the ten-millisecond period. There are thus six measurements, so that each sample can be represented by a point in a six-dimensional space. For each utterance, all amplitude values are normalized so that the highest amplitude found in the first channel has a value of 64. The zero crossing values are proportional to the number of sign changes in the signal in the 10 millisecond period. They are taken as a rough estimate, within each filter band, of the formant location, or the peak position in the energy versus frequency curve.

The 270 utterances (five repetitions of 54 words) resulted in a total of 19,850 sample points. This data was available in digital form on tape.

CLUSTERING

Clustering involves partitioning a set of points in a space into regions according to some criterion of similarity. In the work being reported here, the criterion chosen was geometric, or Euclidean, distance between the sample points. The clustering method applied to speech data retains an important phoneme characteristic in that relatively long periods during which the speech parameters vary slowly provide more points and, therefore, are assumed to be more significant than periods of rapid variations. It was thus expected that sample points would be relatively dense in regions of the measurement space corresponding to periods of slow parameter variations and that similar regions would occur in a number of different utterances. The centers of these regions could then be found by clustering analysis. Since these phoneme-like clusters would be generated algorithmically from the hyperspace points and would be used like phonemes, they were called "Hyperphonemes." The abbreviation "HP" will be used in this report.

The clustering methodology used borrowed heavily from techniques reported in the literature[1, 2, 3, 4, 6, 7]. It will be summarized here and described in more detail in Appendix A. The process begins with the selection of a large number of cluster centers scattered throughout the measurement space. These are components of the "natural" clusters being sought. (The reader will find in the referenced literature attempts to define "cluster" and the "similarity" of or "distance" between points in a cluster.) Cluster boundaries are determined by assigning each point to its nearest center. The center of each cluster is then recalculated. This assignment and center recalculation process is repeated until it converges, as indicated by some measure of cluster compactness. The "natural" clusters are then approached by a process of combining the closest clusters, interspersed with reassignment and center recalculation. Because of these iterative processes, the exact location and number of initial clusters should be uncritical, affecting mainly the subsequent computation time.

An important question concerns where to finally stop the combining processes. Where there is a natural number of clusters, such as in single font character recognition, one might expect the minimum cluster distance to take a sudden jump when the subclusters within the natural ones have all been combined[3]. No such terminating signal appeared for the speech data, so the end point for the cluster combining process was arbitrarily chosen as ten clusters. A large number of clusters provides more resolution for distinguishing among speech utterances at a cost of more segments to store and more computation in the matching process. Fewer clusters means reduced storage and computation but more chance of confusion among acoustically similar utterances.

After the iteration process, the clusters remaining were ordered by magnitude of the amplitude components of the coordinates of the cluster centers and given hyperphoneme identification numbers. The list of the HPs used for subsequent matching is given in Figure 2. In this table, each HP identification number is followed by the HP number of the four closest HPs and the square of the distance separating them from the given HP. The concept of distance between neighboring HPs is crucial to the subsequent process of identifying unknown utterances. The six coordinates of each HP are also given in Figure 2. A1, A2, and A3 are derived from the amplitudes of the signals in the three channels. The Zs are derived from the zero crossing data. The number of points that were assigned to each HP are shown along with the variance calculated as the average squared distance of the points from the cluster center. There were 132 points which did not participate in the final determination of the center locations because they were more distant than 16 from all the cluster centers.

SEGMENTATION

Segmentation involves partitioning an utterance into fundamental units. An unknown utterance will later be identified by matching its segment pattern against a standard representation in a dictionary.

In order to generate the segments, each of the points of the original 270 utterances was labeled with the ID number of the closest HP. Successive points with the same ID number were then grouped into a segment and the segment was assigned a duration equal to ten milliseconds times the number of successive points. Figure 3 shows a time scale plot of the segments for the 5 repetitions of the word DELETE. The HP number is shown for each segment.

COMPACTING

The HPs represent regions in the measurement space. The clustering process should, by definition, minimize the number of sample points near the region boundaries. However, there were cases where the path of successive samples oscillated across a boundary, resulting in several segments whose labels alternated between two HPs. There were also 10 millisecond (one sample point) segments preceded and followed by segments from relatively distant regions. It seemed that both dictionary storage space for the standard representation of each word and computation time for the matching process could be reduced by combining the oscillating sequences and eliminating the short isolated segments. It also appeared that adjacent segments whose HPs were closest neighbors in the measurement space could usually be combined with no loss of discrimination among words.

For compacting and for subsequent matching, a maximum neighbor distance was defined such that two HPs separated by no more than this distance would be called neighbors. The distance was chosen as the smallest that provided each HP with at least one neighbor. From Figure 2, it is seen that HP 6 is farthest from its closest neighbor, requiring a maximum neighbor distance of 24 ($\sqrt{558} = 23.6$). It turned out that two HPs have two neighbors and one has three. The compacting rules caused adjacent neighbor segments to be combined. They are given in Appendix B.

The compacting operation has a price. The information lost can in some cases be crucial to the subsequent matching process. For example, in the word QUARTER, two short HP 7 segments surrounded by two longer HP 9 neighbor segments are combined with them and an interior ten millisecond HP 5 segment is dropped, throwing away the "t" sound and forming one long HP 9 segment, which looks like the long HP 9 segment in CORE. More often the results were to make corresponding segments in repetitions of the same word look more alike, but it is apparent that too much compacting was done.

An example of the results of the compacting operation are shown in Figure 4 for DELETE. HP numbers with a period represent an HP half way between the numbers shown; e.g., HP 5.6 is between HP 5 and HP 6.

MATCHING

At this point, a speech utterance was characterized as a string of number pairs, each pair representing a speech segment and consisting of a 6-bit duration in tens of milliseconds and a 5-bit number. Dictionary entries for reference could thus be stored using one 36-bit computer word for a heading and one word for every three segments. The basic dictionary required an average of 3 1/2 words per entry, compared with about 14 words for comparable information in the Vicens-Reddy system[8]. The next requirement for identifying unknown utterances was an algorithm to provide a numerical measure of the similarity of two utterances characterized as strings of hyperphonemes.

In the comparison between an unknown utterance and a dictionary entry, a penalty score is accumulated for every difference found. A zero score indicates near identity and the dictionary entry with the lowest score is taken as the best match. In contrast to the algorithmic nature of the classification system, the matching rules are quite heuristic. They are probably oriented too closely to the characteristics of the test data and may need to be modified if new data are tried.

The matching algorithm is given in Appendix C. It involves setting up correspondences between segments of two utterances based on their HP neighbor distance. Penalty scores are applied for position, duration and HP number differences between corresponding segments. The component scores in a matching are accumulated as they are calculated. In a comparison of an unknown utterance with a series of dictionary candidates, the matching is terminated whenever the accumulated score exceeds the lowest score previously calculated.

RESULTS AND CONCLUSIONS

The matching algorithm has been tested on the list of 270 utterances. The first repetition of each of the 54 words, the ones used to generate the hyperphonemes, were stored as a dictionary. The remaining 216 utterances were then matched against the dictionary. 189 of these, or 87%, were correctly identified by the lowest score.

Some of the incorrect identifications were due to variations in the 5 repetitions of a word such that the first repetition was not the most representative. Therefore, an experiment was run to see how many of the 216 utterances had to be added to the dictionary to insure 100% correct recognition. Since a few of the correct identifications were by very close margins, a larger score difference was required for acceptance. An utterance was added to the dictionary whenever its lowest score in comparison with a correct entry was not at least ten points below the lowest comparison with all the wrong entries. The first pass resulted in 28 new dictionary entries. Since some of these new entries were too close to previously correctly identified utterances, a second pass was made using the expanded dictionary. Four more utterances were added to the dictionary, resulting in a total of 32 entries in addition to the original 54. At this point the system stabilized since a third pass did not result in any further additions to the dictionary. Figure 5 shows some samples of the printout from the third pass. For each utterance the scores achieved in matching with the correct dictionary entry or entries are shown in comparison with the lowest score achieved in comparison with the wrong entries. Each utterance compared with itself achieved, of course, a 0 score.

It is important that the 216 utterances representing the second through fifth repetitions of each word were not involved in the generation of the hyper-phonemes or of the original dictionary entries. The initial 87% recognition of these 216 was achieved with almost no attempt to optimize rules or parameters. Adding 32 of the 216 to the dictionary resulted in correct identification of all 216. These 32 actually represented 26 different words of which 22 required one extra dictionary entry, 2 required two extra, and 2 required three extra. CORE and FOUR were the worst cases and inspection of the data shows that information lost in the combining process was crucial. CORE was most often confused with QUARTER, and FOUR most often with CORE. Requiring 2 extra entries were WORD and DIVIDE. WORD was confused with FOUR and QUARTER; DIVIDE, with NINE and WORD. A majority of the cases requiring a second dictionary entry involved an excessive variation among the repetitions of a word rather than a very low competing score from some other word.

Comparison with the Vicens-Reddy[8] results is difficult since their results on the same data are presented differently. However, it seems clear to me that their correct identification rate is higher. What the comparison would be with a comparable period of development is impossible to conjecture.

The significance of the work lies in the compact dictionary, the algorithmic nature of the clustering and segmentation, and the neighbor-distance concept used in comparing strings.

TIMING

The work was done on a time-shared PDP-10 at the A.I. Project with about six other users active. The programs were written in SAIL, an ALGOL-like language. It is not possible to estimate the amount of computer time devoted to the computation, so elapsed time estimates are given here.

Not counting manual interactions which could be programmed, it took about 12 minutes to enter 3231 converted and scaled points and arrive at the initial 212 clusters. The time consuming part of the subsequent clustering iterations was the assignment of all points to their closest cluster center. This time depended on the number of clusters remaining after combining closest clusters. 24 assignment cycles were used in going from 212 to 10 clusters. These cycles took about 72 minutes. It is probable that a much smaller number of initial clusters would have worked as well, greatly reducing the calculation time. About 10 minutes were taken by the other operations. Final assignment of all 16000 points to their nearest HP took about 10 minutes. Compacting took about 5 minutes. In matching, each comparison of an utterance with 54 dictionary entries took about 1 second. No attempt was made to implement a candidate selection process to reduce the number of matchings attempted.

DIRECTIONS FOR FUTURE WORK

The most important problem for future work deals with the use of additional test data to determine the extent to which the combining and matching rules have been optimized for the particular set of test data. The next area of concern is parameter and procedure optimization. The only attempt at optimizing parameters occurred when variations were tried on the scores applied to unlinked segments. Some candidates for optimization include the choice of logarithmic conversion and the initial scale factors for the six parameters, the number of hyperphonemes, the rules for combining segments, the maximum distance allowed for neighbors, and the matching rules and parameters. It has been pointed out, for example, that the combining rules for ten millisecond segments sometimes result in throwing away important information.

Finally, it would be interesting to investigate the effects of higher resolution input with more channels.

NOTE: A very relevant reference, by Steingrandt and Yau[9], appeared after completion of this paper. They classify waveforms by sequential features which are like hyperphonemes, although the clustering technique and distance measures are different and segment duration is ignored. They used 50 channels in their input analyzer. Recognition was done by a sequential state machine for each dictionary word, made as a composite of many training samples. Recognition results were presented only for the training set.

APPENDIX A. CLUSTERING METHODS

In clustering points whose individual dimensions are different kinds of measurements, the choice of scaling factors is very important. A change in scaling factor will change the similarity measure (distance in our case) among points, altering the way the points are grouped into clusters. This problem arises with the six dimensions of the speech data, since amplitude and zero crossings are measured in different units and the relative importance of the measurements from the three channels is unknown. The first choice of scale factors gave good enough results so that obvious opportunities for optimization were deferred as candidates for follow-on work.

The fundamental scale factor decision was the choice of a logarithmic conversion, to conform with human perception of relative amplitude and pitch changes. The amplitudes in the second and third (higher frequency) channels were first divided by two in order to give them less weight than the amplitude in the first channel. Since the zero crossing counter in the preprocessor only operates when the signal amplitude is above a preset minimum, zero crossing values corresponding to frequencies well below the filter lower cutoff are sometimes encountered. Such values were arbitrarily set equal to zero for the second and third channel and signal values were converted as ratios to the cutoff value chosen. After logarithmic conversion, further scale factors were applied to all values, equivalent to raising the arguments to various powers. This brought the resulting values into an approximate range of 0-60 for the amplitudes and 0-25 for the zero crossings. As an example, an amplitude of 64 in the second channel would lead to a value of 50, using a scale factor of 33.3:

$$50 = 33.3 \log_{10} \left(\frac{64}{2}\right) .$$

A channel 2 zero crossing value of 24 would lead to:

$$15 = 50 \log_{10} \left(\frac{24}{12}\right)$$

where 12 represents to lower cutoff value and 50, the scale factor. The ranges chosen allow a maximum diagonal across the hyperspace of about 113. The scale factors greatly de-emphasize the importance of the zero crossings and allow all values to be represented by 6-bit numbers. The discontinuity in taking the logarithm of a zero argument was arbitrarily resolved by using the same value as for an argument of one, namely zero. It might have been better to add one unit to all arguments before conversion.

After conversion, some small-valued sample points at the beginning and end of an utterance became zero and were eliminated, leaving about 16,000 points. The points from the first repetition of each of the 54 words, a total of 3,231, were then used for the clustering process. The only difference from standard reported clustering techniques was in the method of forming initial clusters. The basic idea was to form many small clusters in the high density regions. A density was calculated for each sample point as the number of points lying within a hypersphere centered on the given point. The hypersphere radius chosen was eight. (The maximum distance between points was approximately 80.) The points were then ordered by density and the highest density point was taken as the first cluster center. Subsequent cluster centers were formed from the remaining points in descending density order, taking only those points whose distances from all previously selected cluster centers were greater than a selected minimum and whose densities were greater than one. The minimum distance determined the number of initial cluster centers. A radius of eight gave 212 starting clusters. There were 39 isolated points (density = 1) which were not included in any of the initial clusters. An iterative procedure was then followed consisting of the following two steps:

(1) Assign all points to the nearest cluster center, omitting those farther than a predetermined maximum distance from all cluster centers. Recalculate the center positions. Repeat until the number of unassigned points and the mean square distance of all assigned points from their cluster center are stabilized.

(2) Combine the closest clusters so that all remaining clusters are separated by at least a minimum distance. Each combination involves a merger of points and a recalculation of the cluster center. This was done until a 40-50% reduction in the number of clusters was made. Since combining two clusters preserves their outer boundary while merging the center locations, it seemed reasonable to then reassign (step 1). This may be unnecessary until the end of the combination process.

All distances were calculated as the sum of the squares of the individual dimension differences. Since only relative distance comparisons were needed, the square root never had to be computed. To speed up the process of finding all points within a given distance of a given point, an indexing technique was used. This involved partitioning the hyperspace into cubes and chaining together the points within each cube. For distances no greater than the cube side, only the cubes surrounding the given point needed to be searched. This was used in the original density calculations and in the nearest-neighbor assignment process.

APPENDIX B. COMPACTING RULES

The compacting rules for combining adjacent segments were:

(1) Segments not immediately adjacent to a neighbor were not affected.
Example: 4(20), 9(30) → 4(20), 9(30). HP 4 for 20 milliseconds followed by HP 9 for 30 milliseconds remains unchanged.

(2) Two segments with the same HP numbers surrounding a ten-millisecond segment were combined, including in the total duration the extra ten milliseconds.
Example: 4(20), 9(10), 4(30) → 4(60).

(3) Two adjacent neighbor segments were combined. Since HPs 2, 5 and 7 had more than one neighbor, there were some sequences of three or more segments with different HPs wherein all adjacent segments were neighbors. In these cases, the closest neighbor pairs were combined first. Under this rule, a combined segment did not participate in further combinations unless it was originally part of an oscillating sequence of the same two HPs When adjacent segments were combined, their durations were summed. If the duration corresponding to either HP had two or more times the duration corresponding to the other, the HP number representing the longer duration was applied to the combined segment. Otherwise, a number representing an HP half way between the two was applied. For this purpose, numbers 11-17 were added to the HP list.
Examples: 3(30), 4(60) → 4(90). 3 and 4 are neighbors.
3(30), 4(50) → 3.4(80). 3.4 is halfway between 3 and 4. This form is used for printing instead of the actual internal HP number.
4(30), 3(30), 4(40) → 4(100). Oscillating sequence.
5(30), 7(30), 9(30) → 5(30), 7.9(60). 7 is closer to 9 than to 5.

(4) Ten-millisecond segments not covered by the preceding rules were eliminated.
Example: 4(20), 9(10), 7(30) → 4(20), 7(30).

APPENDIX C. MATCHING ALGORITHM

In matching, a numerical measure is sought for the similarity of two utterances.

The matching process begins with a comparison of overall duration. A duration difference greater than 150 milliseconds immediately terminates the matching process with a rejection. Otherwise, duration differences greater than 40 milliseconds receive a score of 0.2 times the excess over 40. This often helps to distinguish among utterances with similar parameters.

When a word is spoken several times at slightly different speeds, we cannot expect all parts of it to be proportionally speeded or slowed. Nevertheless, it proved helpful, in looking for correspondences between parts of two examples of the same word, to make their total durations the same. Therefore, the segment durations of the shorter utterance are expanded proportionally, rounded to the nearest ten milliseconds, so that the overall durations of the two utterances are equal. The new segment boundaries are used for subsequent position correspondence calculations, but the original durations are also retained for duration difference measures. Figure 6 shows an example of the first and fourth versions of DELETE to illustrate matching. Since the overall duration difference is 30 milliseconds, the difference score is zero. Figure 7 shows the conditions after expansion of the first utterance.

The matching algorithm forms links between "corresponding segments" of the two utterances being compared. Corresponding segments are defined roughly as closest neighbors occupying similar time positions in the two utterances. The algorithm must, therefore, consider each segment of each utterance, searching the corresponding position of the other utterance for neighbors. To do this, it considers in time sequence each segment of one utterance and then each segment of the other. For each segment being considered, a region of the other utterance is scanned beginning 30 milliseconds before the start of the segment under consideration and ending 30 milliseconds after the end of the segment under consideration to allow for speed variations. For each segment of which any part is in the scanned area, two measures are calculated. The first measure is the overlap with the original segment under consideration. The overlap is a measure of time position correspondence consisting of the number of ten millisecond periods common to both segments in the expanded plot. If they have no segments in common, the overlap is negative and its magnitude is the number of ten millisecond periods separating them. The second measure is the neighbor distance between the HPs of the two segments being compared. The actual distance is used here if it is no greater than 24, the maximum neighbor distance chosen in the preceding compacting process. A more distant HP is eliminated from consideration. A closeness index is formed consisting of the distance measure less the smaller of ten or two times the overlap measure. The lowest closeness index selects the closest neighbor (if any) within the scanned range, with some bias towards the most overlapping segment in the case of two candidates with nearly equal distance measures. A link is formed from the original segment under consideration to the selected segment in the scanned area. If the two utterances being compared are thought of as laid out on parallel lines as shown in Figure 7 and links are thought of as a line from the middle of one segment to the middle of another, then no link is allowed to cross any previous one. A segment has only one link to another but is allowed up to five links from other segments (a question of storage allocation) as long as these links do not cross other links. Figure 8 shows the example with the links indicated by arrows.

A neighbor distance score is applied once to every linked pair of segments as a measure of how closely they match. This is the distance between their HPs times the actual (not expanded) duration in milliseconds of the shorter segment of the pair, divided by 100. The thought here was that a short segment whose HP is a neighbor of, rather than identical to, that of a longer segment to which it is linked should not be penalized as much as a longer segment would be. The duration difference is separately penalized. In the example, the distance between HP 5 and HP 5.6 is 12. Therefore, the third linked pair gets a score of 13 since the original duration of the HP 5 segment was 110 milliseconds. Segments which are not linked to any other are penalized by a score of 15 plus 3 times the original duration of the segment in tens of milliseconds. This non-linked penalty resulted from a few experiments with different values, the only such experiment made. Figure 9 shows the neighbor distance scores applied to the example.

A difference in duration between linked segments is the main thing distinguishing some correct matches from incorrect ones. Therefore, a duration difference score is calculated. The formula is as follows (where DD is measured in tens of milliseconds based on the durations before expansion):

0 if DD is less than 2;
2(DD-1) for DD from 2 through 6;
10+4(DD-6) for DD from 7 through 9;
22+8(DD-9) for DD greater than 9.

This formula applies a much stronger penalty to large differences than to small ones. It applies directly to individual linked pairs such as the 8 to 8 or the 1 to 1.2 in the example. However, where more than two segments are tied together by a continuous chain of links, a group is formed. The 5, 6, 5 to 5.6 group in the example illustrates this. The durations of the grouped segments in each utterance are summed and the overall difference used in the formula. This avoids penalizing oscillations among neighboring segments that remain after the combining operation. For the illustrated group, the three original durations summed to 19, versus 19 for the HP 5.6 segment, for a duration difference of 0. Note that the unmatched HP 8 segment does not participate in this process.

Finally, a position difference score is calculated for each linked pair based on the difference between the center positions of the expanded segments. Integer division is used in calculating the centers, resulting in rounding down to the next lower integer number of tens of milliseconds. The score is half what the duration difference score would be for the same argument. In the example, the matching HP 8 segments have a position difference of 27-24 = 3 for a score of 2. For groups, the center position is calculated as a center of gravity of the segments involved so that unlinked segments inside the group do not participate. In the illustration, the center position of the HP 5, 6, 5 part of the group is (3×1+5×5+11×16) divided by 19 = 10. This is compared with the value of 11 for the HP 5.6 segment for a difference score of 0.

Figure 10 shows the example with duration and position difference scores.

REFERENCES

1. G. H. Ball & D. J. Hall, "ISODATA, a novel method of data analysis and pattern classification," Stanford Research Institute, Menlo Park, Calif., Tech. Report, April 1965.

2. G. H. Ball, "Data Analysis in the Social Sciences," American Federation of Information Processing Societies Conference Proceedings, Fall Joint Computer Conference, 27, Part 1 (Spartan Books, Washington, D.C., Macmillan, London, 1965), pp. 533-560. (Also printed as: "A Comparison of Some Cluster-seeking Techniques," Technical Report No. RADC-TR-66-514, Rome Air Development Center, Griffiss Air Force Base, New York, November 1966.)

3. R. G. Casey & G. Nagy, "An autonomous reading machine," IEEE Trans. Electronic Computer, Vol. C-17, May 1968; also Research Report RC-1768, IBM Corporation, Yorktown Heights, N. Y., February 1967.

4. T. M. Cover & P. Hart, "The nearest neighbor decision rule," presented at the International Symposium on Decision Theory, 1966; also T. M. Cover & P. E. Hart, "Nearest neighbor pattern classification," IEEE Trans. Information Theory, Vol. IT-13, pp. 21-27, January 1967.

5. B. Gold, "Word Recognition Computer Program," Technical Report 452, Lincoln Laboratories, MIT, Cambridge, Mass., 1966.

6. P. E. Hart, "The Condensed Nearest Neighbor Rule," IEEE Transactions on Information Theory, Vol. IT-14, No. 3, May 1968, pp. 515-516.

7. G. Nagy, "State of the Art in Pattern Recognition," Proceedings of the Institute of Electrical and Electronic Engineers, Vol. 56, No. 5, May 1968.

8. P. Vicens, "Aspects of Speech Recognition by Computer," Computer Science Department of Stanford University, Stanford, California, Memo AI-85, CS 127, April 1969.

9. W. J. Steingrandt & S. S. Yau, "Sequential Feature Extraction for Waveform Recognition," American Federation of Information Processing Societies Conference Proceedings, AFIPS Press, Vol. 36, 1970 Spring Joint Computer Conference, pp. 65-76.

INSERT	NAME
DELETE	END
REPLACE	SCALE
MOVE	CYCLE
READ	SKIP
BINARY	JUMP
SAVE	ADDRESS
CORE	OVERFLOW
DIRECTIVE	POINT
LIST	CONTROL
LOAD	REGISTER
STORE	WORD
ADD	EXCHANGE
SUBTRACT	INPUT
ZERO	OUTPUT
ONE	MAKE
TWO	INTERSECT
THREE	COMPARE
FOUR	ACCUMULATE
FIVE	MEMORY
SIX	BYTE
SEVEN	QUARTER
EIGHT	HALF
NINE	WHOLE
MULTIPLY	UNITE
DIVIDE	DECIMAL
NUMBER	OCTAL

Figure 1: 54 word list

HP#	HP Numbers of 4 Nearest Neighbors and Squared Distance				Center Coordinates--Amplitude and Zero-crossing Data for the 3 channels							Number of Points Defining HP and Mean Square Distance From the Center
						A1	Z1	A2	Z2	A3	Z3	
1	2-369,	3-685,	4-1268,	5-1587,	CENTER IS	Ø	1	1	2	1	5,	41ØPTS, VAR = 5Ø
2	1-369,	5-456,	4-987,	3-113Ø,	CENTER IS	18	5	1	Ø	1	Ø,	389PTS, VAR = 75
3	4-4Ø3,	1-685,	2-113Ø,	5-2298,	CENTER IS	2	Ø	1	2	23	19,	285PTS, VAR = 99
4	3-4Ø3,	6-963,	2-987,	1-1268,	CENTER IS	21	6	1	1	25	2Ø,	27PTS, VAR = 139
5	7-429,	2-456,	6-558,	4-1315,	CENTER IS	38	12	3	1	Ø	1,	433PTS, VAR = 74
6	5-558,	7-633,	8-715,	4-963,	CENTER IS	47	13	6	2	21	6,	111PTS, VAR = 16Ø
7	9-4Ø3,	5-429,	6-633,	8-9Ø8,	CENTER IS	49	15	2Ø	4	1	1,	393PTS, VAR = 11Ø
8	10-324,	6-715,	9-8Ø3,	7-9Ø8,	CENTER IS	55	16	29	1Ø	28	6,	236PTS, VAR = 147
9	7-4Ø3,	1Ø-791,	8-8Ø3,	6-157Ø,	CENTER IS	55	19	38	9	2	Ø,	5Ø1PTS, VAR = 94
10	8-324,	9-791,	7-16Ø8,	6-1941,	CENTER IS	58	2Ø	46	13	28	5,	314PTS, VAR = 89

Figure 2: Hyperphoneme (HP) List

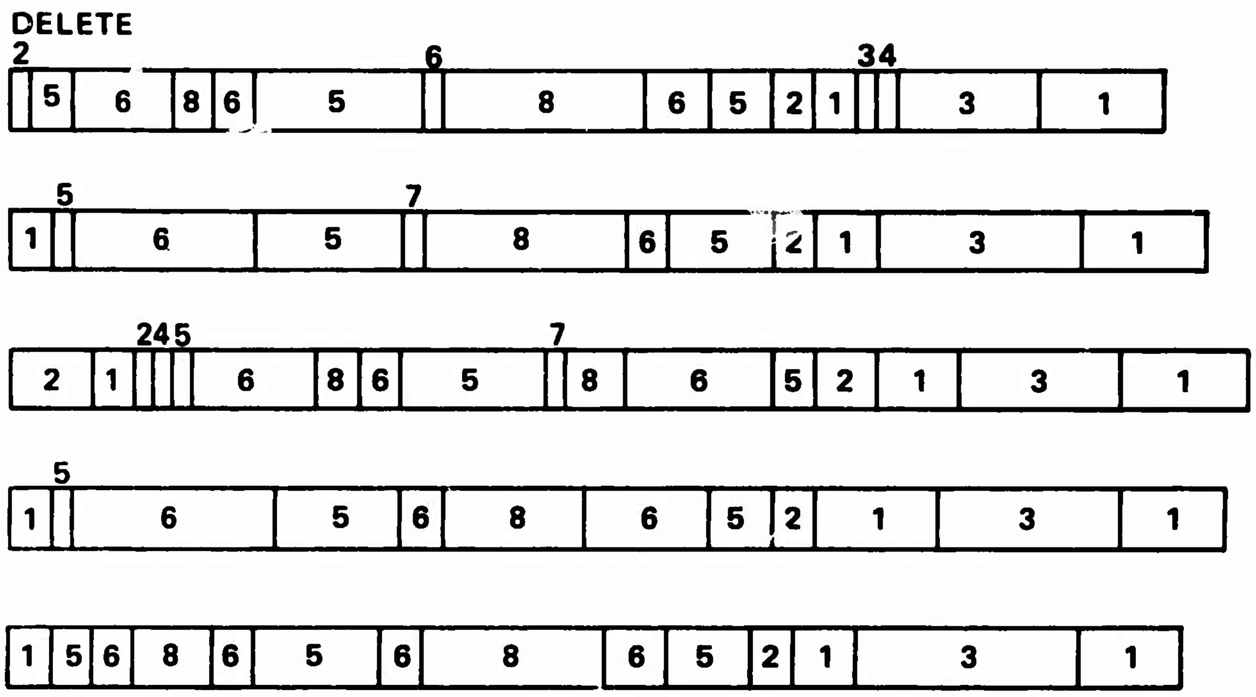

Figure 3. The 5 repetitions of DELETE, illustrating segmentation

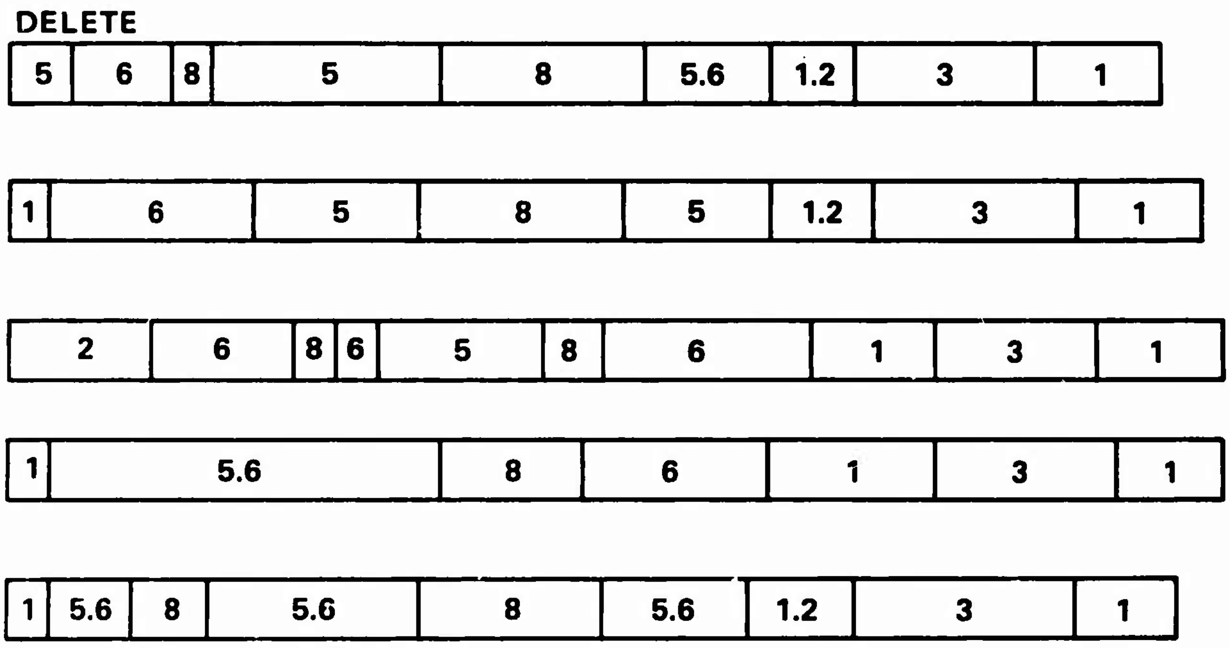

Figure 4. Segments of DELETE after compacting

```
OUTPU       WORD NO. 42
1:  42-Ø      42-72      SMALLEST ERROR SCORE IS  9-169
2:  42-89     42-112     SMALLEST ERROR SCORE IS  47-164
3:  42-63     42-22      SMALLEST ERROR SCORE IS  54-1Ø5
4:  42-1Ø1    42-79      SMALLEST ERROR SCORE IS  54-146
5:  42-135    42-Ø       SMALLEST ERROR SCORE IS  9-1Ø8
2 DICTIONARY ENTRIES

MAKE        WORD NO. 43
1:  43-Ø      43-53      SMALLEST ERROR SCORE IS  48-58
2:  43-53     43-Ø       SMALLEST ERROR SCORE IS  23-55
3:  43-87     43-42      SMALLEST ERROR SCORE IS  23-66
4:  43-77     43-48      SMALLEST ERROR SCORE IS  23-77
5:  43-97     43-43      SMALLEST ERROR SCORE IS  48-1Ø5
2 DICTIONARY ENTRIES

INTER       WORD NO. 44
1:  44-Ø     SMALLEST ERROR SCORE IS  14-271
2:  44-73    SMALLEST ERROR SCORE IS  52-227
3:  44-57    SMALLEST ERROR SCORE IS  52-2Ø6
4:  44-4Ø    SMALLEST ERROR SCORE IS  52-189
5:  44-83    SMALLEST ERROR SCORE IS  14-314
1 DICTIONARY ENTRIES

COMPA       WORD NO. 45
1:  45-Ø     SMALLEST ERROR SCORE IS  37-184
2:  45-5Ø    SMALLEST ERROR SCORE IS  26-17Ø
3:  45-34    SMALLEST ERROR SCORE IS  37-194
4:  45-47    SMALLEST ERROR SCORE IS  47-182
5:  45-59    SMALLEST ERROR SCORE IS  37-166
1 DICTIONARY ENTRIES
```

Figure 5: Sample of Matching Results

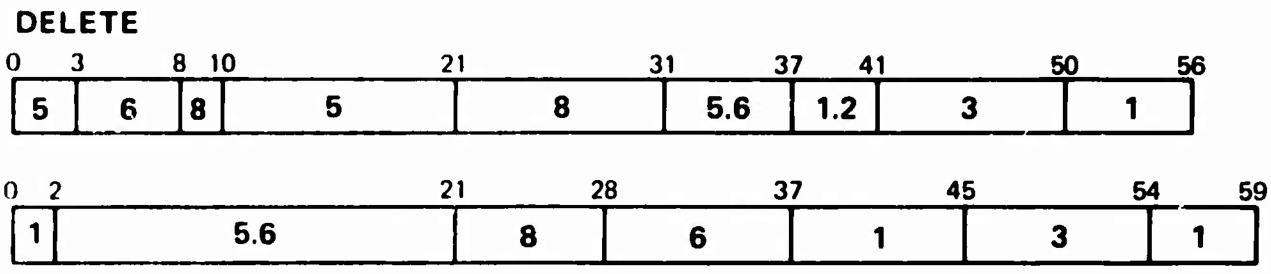

Figure 6. Example to illustrate the matching algorithm

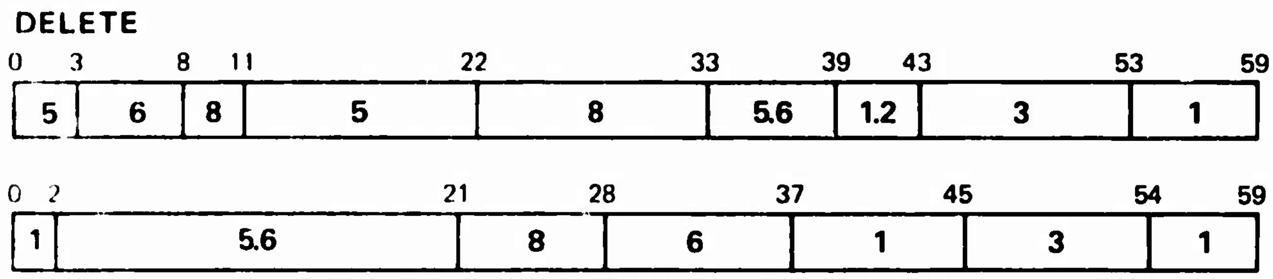

Figure 7. Example after expansion

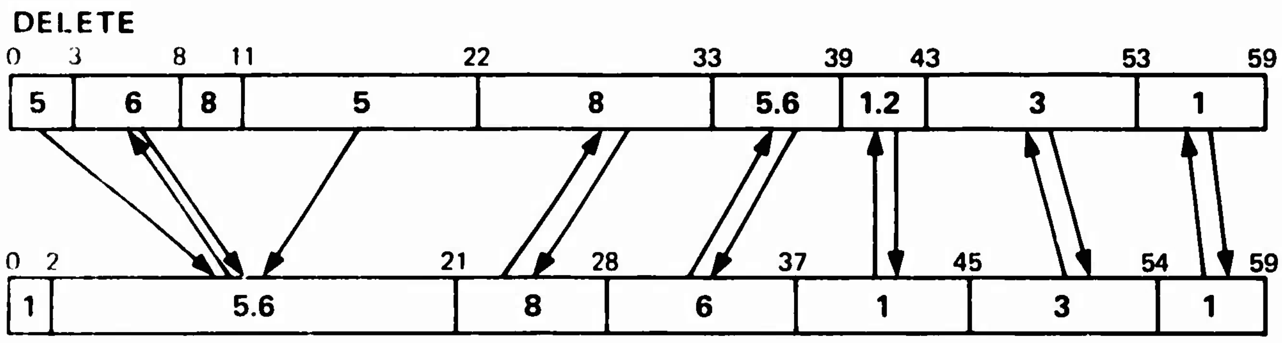

Figure 8. Example with segments linked

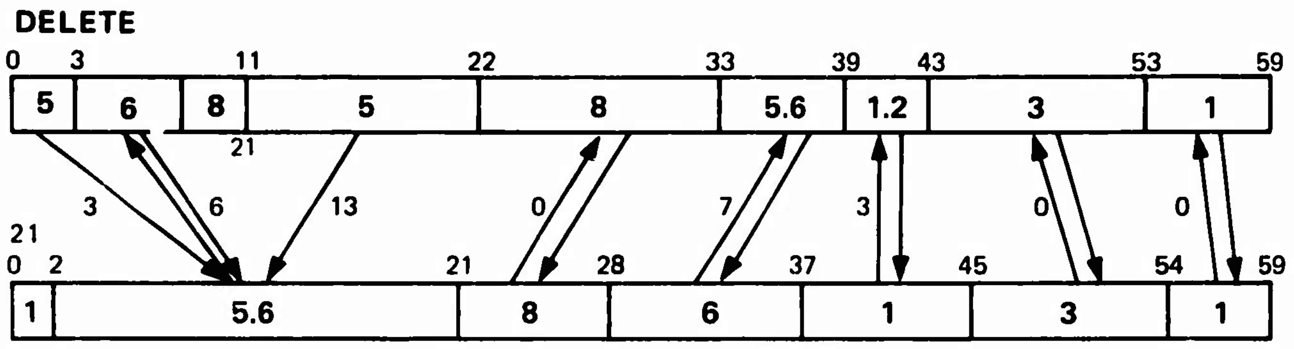

Figure 9. Example with neighbor distance scores

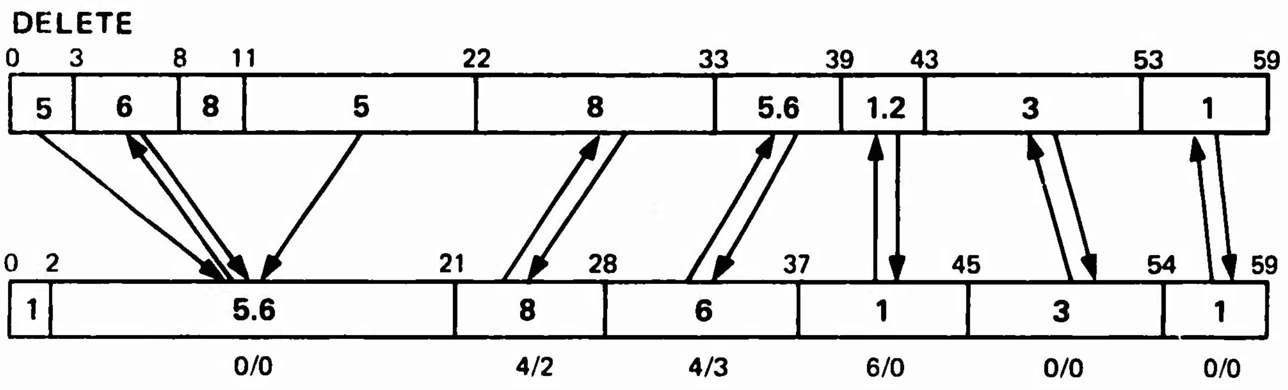

Figure 10. Example with duration/position difference scores